Örebro universitet
Institutionen för teknik
Thomas Padron-McCarthy (Thomas.Padron-McCarthy@tech.oru.se)

Tentamen i Objektorienterad programmering för D2 m fl

tisdag 28 augusti 2007 kl 08:00 - 13:00 i L001 och L003

Gäller som tentamen för:
OBP012 Objektorienterad programmering, provkod 0101 (och 0100)
TDD122 Tillämpad datavetenskap B, provkod 0330 (och 0200)

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 40. För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas på kursens hemsida senast måndag 10 september 2007.
Visning och frågestund:	Tisdag 11 september 2007 kl 12:00-12:30 i mitt rum (T2220). Efter visningen kan tentorna hämtas på expeditionen. Man kan också få sin rättade tenta hemskickad.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070-73 47 013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara och tvetydiga formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

För alla uppgifter på tentan gäller: Man får använda både standard-C++ (som till exempel har pekare som anges med * och objekt som skapas med **new**) och Microsofts dialekt C++/CLI (som till exempel har pekare som anges med ^ och objekt som skapas med **gcnew**). Man kan dock få poängavdrag om man blandar dem på ett felaktigt sätt, till exempel om man skapar ett objekt med **gcnew** och sen pekar på det med en *-pekare.

Uppgift 1 (2 p)

Ett C++-program som ska arbeta med data om rum behöver en klass **Rum**, som ska användas för att representera rum. Ett rum har följande egenskaper:

- en bredd (mätt i meter), till exempel 3.2
- en längd (mätt i meter), till exempel 9.1
- ett namn, till exempel **köket**

Skriv klassdefinitionen för klassen **Rum**. Den ska ha en lämplig konstruktor. Det ska också finnas medlemsfunktioner för att komma åt de olika variablerna

Uppgift 2 (2 p)

Implementera konstruktorn och de övriga medlemsfunktionerna i **Rum**.

Uppgift 3 (4 p)

Vi vill kunna skriva ut rum på standardutmatningen (som normalt är kopplad till skärmen), och även läsa in rum från standardinmatningen (som normalt är kopplad till tangentbordet). Därför lägger vi till två medlemsfunktioner, **skriv** och **las**, och lägger till dem i klassdefinitionen med följande funktionshuvuden:

```
void skriv();
void las();
```

Implementera medlemsfunktionerna **skriv** och **las**.

Uppgift 4 (2 p)

Förutom funktionen **skriv** i uppgiften ovan, vill vi skapa en riktig utskriftsoperator för klassen **Rum**. Vi lägger därför till följande i klassdefinitionen:

```
friend ostream& operator<< (ostream& o, const Rum& rum);
```

Implementera utskriftsoperatorn.

Uppgift 5 (3 p)

Vi ska ha en inläsningsoperator också.

a) Ange vad vi ska lägga till i klassdefinitionen.

b) Implementera inläsningsoperatorn.

Uppgift 6 (3 p)

Vi vill testa klassen **Rum**.

Skriv en **main**-funktion som först skapar ett rum (dvs ett **Rum**-objekt) och sen kollar att de olika medlemsfunktionerna ger rätt resultat. Om alla ger rätt resultat ska programmet skriva ut **Rätt!**, annars **Fel!**.

Därefter ska **main**-funktionen använda **las** för att läsa in data om ett rum, och skriva ut det rummets data med hjälp av **skriv**-funktionen.

I den här och alla andra uppgifter på tentan gäller: Om du ska använda en klass, funktion eller liknande sak från en tidigare uppgift, behöver du inte skriva koden för den på nytt. Du får också använda den även om du inte gjort uppgiften där man skulle skriva den.

Uppgift 7 (3 p)

Skriv ett C++-program som läser in rum från standardinmatningen, och sparar dem på en fil. Välj själv om det ska vara en text- eller binärfil. (Ange vilket du valde!) Välj själv hur användaren ska avsluta inmatningen. (Ange vilket sätt du valde!)

Uppgift 8 (3 p)

Skriv ett C++-program som läser filen med rum från uppgiften ovan, och talar om den sammanlagda ytan för alla rummen på filen.

Uppgift 9 (3 p)

Vi vill kunna lagra rum i länkade listor. Man kan visserligen använda de inbyggda klasser för listor som finns i C++, men i stället vill vi ha en pekare i varje rum som pekar till nästa rum. Därför skapar vi en klass, **LankatRum** ("länkat rum"), som ärver från klassen **Rum**, men som dessutom innehåller en pekare till nästa **LankatRum** i listan.

a) Skriv klassdefinitionen för klassen **LankatRum**. Bestäm själv vilka konstruktorer och medlemsfunktioner som behövs.

b) Implementera alla konstruktorer och medlemsfunktioner.

Uppgift 10 (5 p)

Ett **Hus** består av ett antal rum (av klassen **LankatRum**). Det ska finnas en konstruktor, och medlemsfunktioner för att lägga till ett rum och för att räkna ut husets sammanlagda bostadsyta.

a) Skriv klassdefinitionen för klassen **Hus**.

b) Implementera alla konstruktorer och medlemsfunktioner.

Uppgift 11 (3 p)

Vi vill testa klassen **Hus**.

Skriv en **main**-funktion som skapar ett hus med ett antal rum, och sen skriver ut husets sammanlagda yta.

Uppgift 12 (7 p)

Vi ska skriva följande klasser:

- **Djur**.
- **Fagel**, eller "fågel" som det heter när man kan använda svenska bokstäver. En fågel är en sorts djur.
- **Fisk**. En fisk är en sorts djur.
- **Tiger**. En tiger är en sorts djur, men inte en fågel eller en fisk.
- **Anka**. En anka är en sorts fågel.
- **Struts**. En struts är en annan sorts fågel.

Varje djur har ett **namn**, en **vikt** och en **ålder**. Det ska finnas en virtuell funktion som heter **art**, och som returnerar djurets art, till exempel **tiger**.

a) Skriv alla klassernas klassdefinitioner, och implementera alla konstruktorer och medlemsfunktioner.

b) Skriv en funktion (inte medlemsfunktion) som heter **skriv**, och som skriver ut data (namn, vikt och ålder) om ett djur, vilket som helst. Även djurets art ska vara med i utskriften.

c) Skriv en **main**-funktion som skapar en anka som heter **Kalle**, en struts som heter **Puts** och en tiger som heter **Snuttis**, och som sen skriver ut var och en av dem med hjälp av funktionen **skriv**.